◆用于<...>的函数

```
1  #include <stdio.h>
2  #include <cstdlib>        // 包含 system(),rand(),srand()函数的头文件
3  #include <ctime>          // 用于设置随机种子
4  #include <stdlib.h>       // 需要包含 malloc 的头文件
5  #include <iostream>       // c++库.是 IO 的库
6  #include <stdbool.h>      // bool
7
8  #define MaxSize 50        // 链表最大容限
9
10 using namespace std;       // 调用标准(std)命名空间
11
12 typedef int ElemType;
13
14 typedef struct LNode
15 {
16     ElemType data;
17     struct LNode *next;
18 } LinkNode;
19
20 void Menu();                                          // 控制面板
21
22 void InitList(LinkNode *&L);                         // 创建空表
23 void DestroyList(LinkNode *&L);                      // 销毁表
24
25 bool CreateList(LinkNode *&L, ElemType a[], int n);  // 输入表
26 void DispList_1(LinkNode *L);                        // 输出表
27 void DispList_2(LinkNode *L);                        // 输出表
28
29 void Search_sys(LinkNode *L);                        // 查找系统
30 bool ListEmpty(LinkNode *&L);                        // 判断是否为空
31  int ListLength(LinkNode *L);                        // 返回表长度
32 bool GetElem(LinkNode *L, int i, ElemType &e);      // 按序查找
33  int LocateElem(LinkNode *L, int Bp, ElemType e);    // 按值查找
34
35 void upDate_sys(LinkNode *&L);                       // 修改系统
36 bool ListUpdate(LinkNode *&L, int i, ElemType e);    // 修改元素
37 bool ListInsert(LinkNode *&L, int i, ElemType e);    // 插入元素
38 bool ListDelete(LinkNode *&L, int i, ElemType &e);   // 删除数据
39
40 void main2(LinkNode *&L1, LinkNode *&L2);            // 附加系统
41 void ListReverse(LinkNode *&L);                      // 逆置单链表
```

```
42 void Sort(LinkNode *&L);                                  // 排序表
43 void ListUnion(LinkNode *&L1, LinkNode *&L2);            // 合并有序表
44
45  int input(char c);                                      // sys 选择
46 bool Input(const char *s, int *n);                       // 整型验证
47 void clearInputBuffer();                                 // 清除缓冲区
48
49 int main()
50 {
51
52     int choice=0;
53     char c='N';
54     LinkNode *L;          // 表 1
55     InitList(L);
56
57     LinkNode *L1;         // 表 2
58     InitList(L1);
59
60     srand(time(NULL)); // 生成随机数种子
61
62     do
63     {
64         choice=0; c='N';
65         Menu();
66         cin>>(c);
67         choice=input(c);
68         clearInputBuffer();
69
70         switch(choice)
71         {
72             case 0: {           // ** 退出指令 **
73                 DestroyList(L);
74                 DestroyList(L1);
75                 cout<<"\n 表已销毁。\n";
76                 cout<<"\n 系统已退出。\n";
77                 break;
78             }
79
80             case 1: {           // ** 建立实验表 1 **
81                 int n=0,i;
82                 bool isOK=0;
83                 DestroyList(L);                    // 避免野节点
84                 InitList(L);
85
```

```
86                  if(!Input("\n 请输入表 1 的长度:",&n)) break;
87
88                  n<0?n=0:n;                              // 限制长度
89                  n>MaxSize?n=MaxSize:n=n;
90
91                  ElemType *a = new ElemType[n];          // 动态分配数组
92
93                  for(i=0;i<n;i++)
94                      a[i]=rand(); //生成试验数据
95
96                  isOK=CreateList(L,a,n);
97
98                  printf("\n---- 顺序表 1 建立%s! ----",isOK?"成功":"失败");
99                  delete[] a;  // 释放数组
100                 break;
101             }
102
103             case 2: {           // ** 查找系统 **
104                 Search_sys(L);
105                 break;
106             }
107
108             case 3: {           // ** 修改系统 **
109                 upDate_sys(L);
110                 break;
111             }
112
113             case 4: {           // ** 附加实验 **
114                 main2(L,L1);
115                 break;
116             }
117             default: cout<<"\n---- 输入错误! ----\n";
118         }
119     ////////////////////////////////////
120         cout<<"\n\n 按任意键继续....";
121         c=getchar();
122     } while(choice);
123
124     return 0;
125 }
126
127 ////////////////////////////////////////////////////////////////////////////////////////////////////
128 // ** 控制面板 **
129
```

```
130 void  Menu()
131 {
132     system("cls");
133     cout<<"*******整形链表操作*******\n";
134     cout<<"------------------------\n";
135     cout<<"         1   建立实验表 1        \n";
136     cout<<"         2    查询系统           \n";
137     cout<<"         3    修改系统           \n";
138     cout<<"------------------------\n";
139     cout<<"         4    附加功能           \n";
140     cout<<"------------------------\n";
141     cout<<"         0    退出程序           \n";
142     cout<<"------------------------\n";
143     cout<<"**************************\n";
144     cout<<"请选择：";
145 }
146
147 ////////////////////////////////////////////////////////////////////////////////////////////////////
148 // ** 创建空表 **
149
150 void  InitList(LinkNode  *&L)
151 {
152     L=(LinkNode  *)malloc(sizeof(LinkNode));
153     L->data=0;  L->next=NULL;
154 }
155
156 ////////////////////////////////////////////////////////////////////////////////////////////////////
157 // ** 销毁表 **
158
159 void  DestroyList(LinkNode  *&L)
160 {
161     LinkNode  *pre=L,  *p=L->next;
162     while(p!=NULL)
163     {
164         free(pre);
165         pre=p;  p=pre->next;
166     }
167     free(pre);
168 }
169
170 ////////////////////////////////////////////////////////////////////////////////////////////////////
171 // ** 判断是否为空 **
172
173 bool  ListEmpty(LinkNode  *&L)
```

```
174 {
175     return(L->next==NULL);
176 }
177
178 ////////////////////////////////////////////////////////////////////////////////////////////////////
179 // ** 返回表长度 **
180
181 int ListLength(LinkNode *L)
182 {
183     int n=0;
184     LinkNode *p=L;
185     for(n=0;p->next!=NULL;n++)
186         p=p->next;
187     if(L->data!=n) return -1;   // -1 表征节点丢失
188     return n;
189 }
190
191 ////////////////////////////////////////////////////////////////////////////////////////////////////
192 // ** 输入表 **
193
194 bool CreateList(LinkNode *&L, ElemType a[], int n)
195 {
196     if(L->data!=0)  return false;
197     LinkNode *s, *r=L;
198     for(int i=0;i<n;i++)
199     {
200         s=(LinkNode *)malloc(sizeof(LinkNode));  // 申请节点
201         s->data=a[i];
202         r->next=s;       // 尾节点 r 作为链接对象
203         r=s;          // 移动 r
204     }
205     r->next=NULL;      // 处理尾节点
206     L->data=n;         // 将表长存入头节点
207     return true;
208 }
209
210 ////////////////////////////////////////////////////////////////////////////////////////////////////
211 // ** 输出表 **
212
213 void DispList_1(LinkNode *L)
214 {
215     LinkNode *p=L->next;
216     printf("\n- - - - - - - - - - - - - - - -\n\n");
217     for(int i=0; i<L->data; i++, p=p->next) {
```

```
218             printf("%6d",p->data);    printf("   -> ");
219             if((i+1)%3==0)  printf("\n");
220         }
221         printf("   NULL");
222         printf("\n_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _\n");
223 }
224
225 void  DispList_2(LinkNode  *L)
226 {    // 显示所有元素与其序号的对应关系
227         LinkNode  *p=L->next;
228         cout<<"\n 当前表中的元素及其序号对映：\n";
229         cout<<"_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _\n";
230         for(int  i=1;  i<=L->data;  i++,  p=p->next)
231             if(p!=NULL)
232                 cout<<"   序号  "<<i<<":  \t\t"<<p->data<<endl;
233         cout<<"_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _\n";
234 }
235
236 ////////////////////////////////////////////////////////////////////////////////////////////////////////////
237 // **  查找系统  **
238
239 void  Search_sys(LinkNode  *L)
240 {
241         int  choice;
242         char  c='N';
243          do  {
244             choice=999;  c='N';
245             system("cls");
246             cout<<"*******整形链表操作*******\n";
247             cout<<"------------------------\n";
248             cout<<"---------查询系统---------\n";
249               cout<<"          1   返回表长度          \n";
250               cout<<"          2   遍历显示表          \n";
251               cout<<"          3  按序查找元素         \n";
252               cout<<"          4  按值查找元素         \n";
253               cout<<"          0     退出程序          \n";
254               cout<<"------------------------\n";
255             cout<<"**************************\n";
256               printf("\n 请选择  (0-4):  ");
257               cin>>(c);
258               choice=input(c);
259               clearInputBuffer();
260
261               switch(choice)  {
```

```
262             case 0: return;// 退出子程序
263
264             case 1: {
265                 int n=ListLength(L);
266                 if(n==-1)
267                     printf("\n---- 顺序表节点丢失! ----");
268                 else
269                     printf("\n---- 顺序表长度为%d ----",n);
270                 break;
271             }
272
273             case 2: {
274                 DispList_1(L);
275                 break;
276             }
277
278               case 3: {
279                 int i; bool bl;
280                 ElemType e;
281
282                 if(!Input("\n 请输入要查找的序号:",&i)) break;
283
284                 bl=GetElem(L, i, e);
285                 printf("\n----- %s 该序号 -------\n",bl?"查到":"未查到");
286                 //bl?printf("值为:%d",e):0;
287                 if(bl)    printf("值为:%d",e);
288                 break;
289             }
290
291               case 4: {
292                 int Bp=0,count=0;
293                 ElemType e;
294
295                 if(!Input("\n 请输入要查找的元素值:",&e)) break;
296
297                 cout<<"\n\n-------------------------\n\n";
298                 do {
299                         Bp=LocateElem(L,Bp,e);
300                         count++;
301                         if(Bp!=0) printf("第%d 个匹配元素的位置: %d\n", count, Bp);
302                 } while (Bp);
303                 if(count==1) printf("未找到值为 %d 的元素位置!\n",e);
304                 cout<<"\n-------------------------\n";
```

```
305                     break;
306                 }
307                   default: cout<<"\n---- 输入错误！ ----\n";
308             }
309     ////////////////////////////////////
310         cout<<"\n\n 按任意键继续....";
311         c=getchar();
312      } while (choice!=0);
313 }
314
315 // ** 按序查找 **
316
317 bool GetElem(LinkNode *L, int i, ElemType &e)
318 {
319     LinkNode *p=L;
320     if(i<=0) return false;
321     for(int j=0; j<i && p!=NULL; j++, p=p->next);
322     if(p==NULL) return false;
323     e=p->data;   return true;
324 }
325
326 // ** 按值查找 **
327
328 int LocateElem(LinkNode *L, int Bp, ElemType e)
329 {                       // Breakpoint 查询断点
330     int i; LinkNode *p=L->next;
331     for(i=Bp; i>0; i--)
332         p=p->next;
333     for(i=Bp; i<L->data; i++, p=p->next)
334         if(p->data==e)
335             return i+1;
336     return 0;
337 }
338
339 ////////////////////////////////////////////////////////////////////////////////////////////////////
340 // ** 修改系统 **
341
342 void upDate_sys(LinkNode *&L)
343 {
344     int choice;
345     char c='N';
346      do {
347         choice=999; c='N';
348         system("cls");
```

```
349          cout<<"*******整形链表操作*******\n";
350          cout<<"-------------------------\n";
351            cout<<"---------修改系统---------\n";
352            cout<<"          1  定点插入元素          \n";
353            cout<<"          2  定点修改元素          \n";
354            cout<<"          3  查值修改元素          \n";
355          cout<<"          4  定点删除元素          \n";
356          cout<<"          5  查值删除元素          \n";
357            cout<<"          0     退出程序           \n";
358            cout<<"-------------------------\n";
359          cout<<"*************************\n";
360            printf("\n 请选择  (0-5):  ");
361            cin>>(c);
362            choice=input(c);
363            clearInputBuffer();
364
365            switch(choice)  {
366               case  0:  return; // 退出子程序
367
368               case  1:  {
369                    int  i;
370                    ElemType  e;
371
372                    if(ListEmpty(L)==false)
373                    {
374                         DispList_2(L);
375                    }  else  {
376                         printf("\n 表为空,请注意只能在首位插入!!!\n");
377                    }
378
379                    printf("\n  请输入插入点  (1-%d)  :",L->data>=MaxSize?MaxSize:
L->data+1);
380
381                    if(!Input("  ",&i))  break;
382
383                    if(!Input("\n 请输入元素:",&e))  break;
384
385                    printf("\n----  插入操作%s  ----\n",ListInsert(L,i,e)?"成功！":"失败
");
386                    printf("\n  此时顺序表长度为  %d  /  %d  ",ListLength(L),MaxSiz
e);
387                    break;
388               }
389
```

```
390                 case 2: {
391                     int i;
392                     ElemType e;
393
394                     if(ListEmpty(L)==false)
395                     {
396                         DispList_2(L);
397                     } else {
398                         printf("\n 表为空,修改操作已退出!\n");
399                         break;
400                     }
401
402                     printf("\n 请输入修改点 (1-%d) :",L->data);
403
404                     if(!Input("",&i)) break;
405
406                     if(!Input("\n 请输入元素:",&e)) break;
407
408                     printf("\n---- 修改操作%s ----\n",ListUpdate(L,i,e)?"成功! ":"失
    败");
409                     printf("\n 此时顺序表长度为 %d / %d ",ListLength(L),MaxSiz
    e);
410                     break;
411                 }
412
413                 case 3: {
414                     int Bp=0,count=0;
415                     char c='N';
416                     ElemType e,e1;
417
418                     if(!Input("\n 请输入要查找的元素值:",&e)) break;
419
420                     cout<<"\n\n-------------------------\n\n";
421                     while(1)
422                     {
423                         c='N'; e1=e;
424                         Bp=LocateElem(L,Bp,e);
425
426                         if(Bp!=0) {
427                             count++;
428                             printf("第%d 个匹配元素的位置:%d\n", count, Bp);
429                         } else {
430                             cout<<"\n-------- 表已查完 --------\n";
431                             break;
```

```
                    }

                    cout<<"\n 是否修改该处的值? ( Y or N ):";
                    cin>>(c); clearInputBuffer();

                    if(c=='y' || c=='Y')
                    {
                        if(!Input("\n 请输入元素:",&e1)) break;
                        printf("\n ---- 修改操作%s ----\n",ListUpdate(L,Bp,e1)?"成功! ":"失败");
                    }

                    cout<<"\n 是否继续查找该值? ( Y or N ):";
                    cin>>(c); clearInputBuffer();

                    if(c=='n' || c=='N') break;
                    cout<<"\n------------------------\n\n";
                }
                if(count==0)
                    printf(" 未找到值为 %d 的元素位置!\n",e);
                else
                    printf("\n    共找到 %d 个匹配元素.\n",count);
                cout<<"\n------------------------\n";
                break;
            }

            case 4: {
                int i; c='N';
                bool isOK=0;
                ElemType e;

                if(ListEmpty(L)==true)
                {
                    printf("\n 当前表为空,删除操作已退出!");
                    break;
                }

                DispList_2(L);

                printf("\n 请输入删除点: (1-%d) :",L->data);
                if(!Input(" ",&i)) break;

                cout<<"\n 要删除的元素为:";
                printf("%d\n",GetElem(L,i,e)?e:0);
```

```
475
476                 cout<<"\n 是否确认删除? ( Y or N ):";
477                 cin>>(c); clearInputBuffer();
478
479                 if(c=='Y' || c=='y')
480                 {
481                     isOK=ListDelete(L,i,e);
482                     printf("\n---- 删除操作%s ----\n",isOK?"成功! ":"失败!!!");
483                     if(isOK) printf("\n 顺序表长度为 %d/%d \n",ListLength(L),
MaxSize);
484
485                 } else {
486                     cout<<"\n---- 删除操作已取消! ----";
487                 }
488                 break;
489             }
490
491             case 5: {
492                 int Bp=0,count=0;
493                 char c='N';
494                 bool isOK=0;
495                 ElemType e,e1;
496
497                 if(!Input("\n 请输入要查找的元素值:",&e)) break;
498
499                 cout<<"\n\n-------------------------\n\n";
500                 while(1)
501                 {
502                     c='N'; isOK=0;
503                     Bp=LocateElem(L,Bp,e);
504
505                     if(Bp!=0) {
506                         count++;
507                         printf("第%d 个匹配元素的位置:%d\n", count, Bp);
508                     } else {
509                         cout<<"\n-------- 表已查完 --------\n";
510                         break;
511                     }
512
513                     cout<<"\n 是否删除该处的值? ( Y or N ):";
514                     cin>>(c); clearInputBuffer();
515
516                     if(c=='y' || c=='Y')
517                     {
```

```
518                             isOK=ListDelete(L,Bp,e);
                                printf("\n ---- 删除操作%s ---- \n",isOK?"成功！":"
519 失败!!!");
                                if(isOK) printf("\n 顺序表长度为 %d/%d \n",ListLe
520 ngth(L),MaxSize);
521                         }
522
523                         cout<<"\n 是否继续查找该值? ( Y or N ):";
524                         cin>>(c); clearInputBuffer();
525
526                         if(c=='n' || c=='N') break;
527                         cout<<"\n-------------------------\n\n";
528                 }
529                 if(count==0)
530                     printf(" 未找到值为 %d 的元素位置!\n",e);
531                 else
532                     printf("\n    共找到 %d 个匹配元素.\n",count);
533                 cout<<"\n-------------------------\n";
534                 break;
535             }
536               default: cout<<"\n---- 输入错误! ----\n";
537         }
538     ////////////////////////////////////
539         cout<<"\n\n 按任意键继续....";
540         c=getchar();
541     } while (choice!=0);
542 }
543
544 // ** 修改元素 **
545
546 bool ListUpdate(LinkNode *&L, int i, ElemType e)
547 {
548     LinkNode *p=L->next;
549     if(i<1 || i>L->data || i>MaxSize)
550         return false;
551     while(--i>0)      // 移动到节点 i
552         p=p->next;
553     p->data=e;
554     return true;
555 }
556
557 // ** 插入元素 **
558
559 bool ListInsert(LinkNode *&L, int i, ElemType e)
```

```
{
    LinkNode *p=L, *s;
    if(i<1 || i>L->data+1 || L->data>=MaxSize)
        return false;
    while(--i>0 && p!=NULL)
        p=p->next;
    if(p==NULL && i!=L->data+1)return false;
    s=(LinkNode *)malloc(sizeof(LinkNode));
    s->data=e;
    s->next=p->next;     // p->next 连到 s 后
    p->next=s;           // p->next 置为 s
    L->data++;
    return true;
}

// ** 删除数据 **

bool ListDelete(LinkNode *&L, int i, ElemType &e)
{
    LinkNode *p=L, *q;
    if(i<1 || i>L->data)
        return false;
    while(--i>0 && p!=NULL)
        p=p->next;
    if(p==NULL || p->next==NULL)
        return false;
    q=p->next;
    e=q->data;
    p->next=q->next;
    free(q);
    L->data--;
    return true;
}

////////////////////////////////////////////////////////////////////////////////////////////////////
// ** 选做实验 **

void main2(LinkNode *&L1, LinkNode *&L2)
{
    int choice;
    char c='N';
     do {
        choice=999; c='N';
        system("cls");
```

```
604         cout<<"*******整形链表操作*******\n";
605         cout<<"-------------------------\n";
606           cout<<"----------附加实验--------\n";
607           cout<<"          1   逆置单链表          \n";
608         cout<<"          2   建立实验表 2        \n";
609         cout<<"          3   有序表的并          \n";
610           cout<<"          0    退出程序           \n";
611           cout<<"-------------------------\n";
612         cout<<"**************************\n";
613           printf("\n 请选择  (0-3):  ");
614           cin>>(c);
615           choice=input(c);
616           clearInputBuffer();
617
618           switch(choice)  {
619             case  0:  return; // 退出子程序
620
621             case  1:  {
622                 if(!ListLength(L1))
623                 {
624                     cout<<"\n     ---  表为空!  ---";
625                     break;
626                 }
627                 cout<<"\n 逆置前结果:";
628                 DispList_1(L1);
629                 ListReverse(L1);
630                 cout<<"\n 逆置后结果:";
631                 DispList_1(L1);
632                 break;
633             }
634
635             case  2:  {
636                 int  n=0,i;
637                 bool  isOK=0;
638                 DestroyList(L2);                    // 避免野节点
639                 InitList(L2);
640
641                 if(!Input("\n 请输入表  2  的长度:",&n))  break;
642
643                 n<0?n=0:n;                          // 限制长度
644                 n>MaxSize?n=MaxSize:n=n;
645
646                 ElemType  *a  =  new  ElemType[n];          // 动态分配数组
647
```

```
                for(i=0;i<n;i++)
                    a[i]=rand(); //生成试验数据

                isOK=CreateList(L2,a,n);

                printf("\n---- 顺序表 2 建立%s! ----",isOK?"成功":"失败");
                delete[] a;  // 释放数组
                break;
            }

            case 3: {
                int overflow=0;      // 判断是否溢出
                if(ListEmpty(L2)!=0)
                {
                    cout<<"\n---- 表 2 为空,程序已退出! ----";
                    break;
                }

                if(ListEmpty(L1) || L1->data>=MaxSize)
                {
                    cout<<"\n ---- 表 1 为空或已满! ----";
                    break;
                 }

                overflow=L1->data+L2->data;

                Sort(L1);
                Sort(L2);

                cout<<"\n\n  ------- 两表已排序! ------\n";
                cout<<"\n 表 1 排序如下:\n";
                DispList_1(L1);
                cout<<"\n 表 2 排序如下:\n";
                DispList_1(L2);

                ListUnion(L1,L2);   // 调用合并函数

                cout<<"\n\n  ------- 两表已合并! ------\n";
                cout<<"\n 表 1 结果如下:\n";
                DispList_1(L1);
                printf("\n 顺序表 1 长度为 %d/%d \n", ListLength(L1), MaxSi
ze);
                printf("\n  -------- 运算%s --------", overflow>MaxSize?"溢出":
"未溢出");
```

```
690
691                     InitList(L2);      // 重建表 2 以便继续运行程序
692                     break;
693                 }
694                 default: cout<<"\n      ---- 输入错误! ----\n";
695             }
696         ////////////////////////////////////
697             cout<<"\n\n 按任意键继续....";
698             c=getchar();
699         } while (choice!=0);
700 }
701
702 // ** 逆置单链表 **
703
704 void ListReverse(LinkNode *&L)
705 {   // 原地置换法
706     LinkNode *p=L->next, *q=p->next;
707     while(p->next!=NULL)
708     {
709         p->next=q->next;    // 链接 q 后的节点(不丢节点)
710         q->next=L->next;    // 链接 p 前的节点(不丢节点)
711         L->next=q;          // 将 q 插入表头  (逆置)
712         q=p->next;          // 将 q 移至 p 后 (移动)
713     }
714 }
715 /*
716 void InvertLinkedList(LinkNode &L)
717 {   // 头插法
718     LinkNode *p, *q=L->next;
719     L->next=NULL;
720     while(q)
721     {
722         p=q;
723         q=q->next;
724         p->next=L->next;
725         L->next=p;
726     }
727 }
728 */
729 // ** 单链表的排序 **
730
731 void Sort(LinkNode *&L)
732 {
733     if(L==NULL || L->next==NULL) return;
```

```
734
735     LinkNode *p, *q, *end = NULL;   // end 记录已排序部分的边界
736     ElemType ts;
737
738     for(p=L->next; p->next!=NULL; p=p->next, end=q)
739         for(q=L->next; q->next!=end; q=q->next)
740             if(q->data > q->next->data)        // 冒泡
741             {
742                 ts = q->data;
743                 q->data = q->next->data;
744                 q->next->data = ts;
745             }
746 }
747
748 // ** 单链表的并 **
749
750 void ListUnion(LinkNode *&L1, LinkNode *&L2)
751 {
752     if(L1==NULL || L2==NULL) return;
753     LinkNode *p1=L1->next;
754     LinkNode *p2=L2->next;
755     LinkNode *p=L1;
756     while(p1!=NULL && p2!=NULL)
757     {
758         if(p1->data < p2->data)
759             {p->next=p1; p=p1; p1=p1->next;}
760         else
761             {p->next=p2; p=p2; p2=p2->next;}
762     }
763     if(p1)   p->next=p1;
764     else     p->next=p2;
765     L1->data+=L2->data;    // 记录长度
766     L2->next=NULL;         // 释放 L2
767     DestroyList(L2);
768 }
769
770 ////////////////////////////////////////////////////////////////////////////////////////////////////
771 // ** sys 选择验证 **
772
773 int input(char c)
774 {
775     if('0'<=c && '9'>=c)
776         return c-'0';
777     else
```

```
778             return -1;
779 }
780
781 ////////////////////////////////////////////////////////////////////////////////////////////////////
782 // ** 整型输入验证 **
783
784 bool Input(const char *s, int *n)
785 {
786     if(s) printf("%s",s);
787      if(scanf("%d",n)!=1)
788     {
789             printf("\n 输入字符非法!");
790             clearInputBuffer();
791             return 0;
792      }
793      clearInputBuffer();
794      return 1;
795 }
796
797 ////////////////////////////////////////////////////////////////////////////////////////////////////
798 // ** 清除缓冲区 **
799
800 void clearInputBuffer() {
801      while(getchar()!='\n');
802 }
803
```